<Internal Application as Filed>

明細書 18頁
請求の範囲 2頁 } 全21頁
要約書 1頁

明　細　書

セキュリティホール診断システム

技術分野

本発明は、コンピュータのセキュリティホールの有無を診断するシステムに関するものである。

背景技術

図９は特開２００１－３３７９１９（第４－８頁、図３、図４、図１４）に代表される、従来のセキュリティホール診断システムを示す構成図である。従来のシステムは操作装置９００と検査実行装置９０７で構成され、操作装置９００はディスプレイ９０２、画面生成部９０３、操作制御部９０５、表示名定義ファイル９０４及び手順定義ファイル９０６で構成されている。

また、検査実行装置９０７は、実行制御部９０８、対象ホスト情報格納部９０９、複数の検査実行部９１１、及び検査実行手段格納部９１０で構成されている。

図１０は同システムにおける手順定義ファイル９０６の例を示すものである。手順定義ファイル９０６には、検査実行手段９１１の分類キー名と、分類キーとして指定された検査実行手段９１１のプロパティの値毎に表示名、実行タイプ、説明文が記載されている。

図１１は同システムにおける検査実行手段９１１の情報（検査実行情報）を表すものである。検査実行情報には、各検査実行手段９１１を特徴付ける値（プロパティ）がキー名（プロパティ名）に関連付けられて格納されている。つまり、検査実行情報（検査実行手段の情報）は、検

査実行手段に一つずつ含まれており、その検査実行実行手段を特徴付ける情報（＝プロフィール）である。検査実行情報には、複数の項目（プロパティ）を記述することができる。各項目は、プロパティ名で区別される。

次に従来のシステムの動作について説明する。操作装置９００が検査実行装置９０７に接続されると、操作装置９００は表示名定義ファイル９０４及び手順定義ファイル９０６をロードする。

次に、検査実行装置９０７中の検査実行手段格納部９１０に蓄積されている検査実行手段９１１一つ一つから検査実行情報を取り出し、手順定義ファイル９０６に指定されたキー名に対応するプロパティをもとに、各検査実行手段９１１を手順定義ファイル９０６記載のカテゴリに分類する。最後に分類された検査実行手段９１１の一覧をカテゴリ毎にディスプレイ９０２に表示する。

使用者１０１はディスプレイ９０２に表示されたカテゴリを選択し、実行に必要なパラメータを入力し、検査実行を要求する。パラメータの説明は表示名定義ファイル９０４に記載されている情報が利用される。検査実行を要求された操作装置９００は、そのカテゴリに分類された検査実行手段９１１を実行するように、操作制御部９０５を通じて検査実行装置９０７に要求する。

検査実行装置９０７は指定された検査実行手段９１１を呼び出し、その結果、検査のためのパケットが検査対象ホストコンピュータ１０７に送信される。

なお、各検査実行手段９１１は、対象ホスト情報格納部９０９に情報を格納することができ、格納された情報は、他の検査実行手段９１１によって参照することができる。また、使用者１０１が操作装置９００を通じて直接対象ホスト情報格納部９０９に情報を格納することも可能で

ある。

以上が従来のシステムにおける検査の流れである。ここで、カテゴリの表示順序は、手順定義ファイル９０６に記載されている順序であり、これを一般的な攻撃の手順に沿うようにすることで、使用者１０１はディスプレイ９０２に表示された順に検査を行うことによって攻撃者を模擬した検査を行うことができる。

以上のように、従来のセキュリティホール診断システムは、複数の検査実行手段を有し、それらを手順定義ファイルで与えられた方法で分類・表示し、カテゴリ毎に使用者が選択することでそのカテゴリに属する検査実行手段を実行するというものであり、また検査実行手段は直接検査対象ホストコンピュータに対し検査を実行するというものであった。そのため、以下のような問題があった。

各カテゴリ毎に入力しなければならない実行パラメータは、使用者が前の検査の結果から入力しなければならず、あるカテゴリの検査の結果と次のカテゴリへの入力との関係を使用者は理解している必要がある。そのために、使用者はセキュリティ上の知識を必要とされた。

定義ファイルは順次実行のシナリオしか表現できないが、実際の攻撃者は、前に行った攻撃の結果に応じて次に実施すべき攻撃の種類を変化させる場合も多い。従来のシステムでは、次にどのカテゴリの検査を実行するかの判断は使用者が行わなければならず、ここでも使用者にセキュリティ上の知識を必要とした。

攻撃者は、ある目的を持って複雑なステップで構成される攻撃を行う。その一連の攻撃は、さらに大きな目的を達成するための攻撃シナリオの一ステップに過ぎない場合も想定される。従来のシステムでは、このような階層化された攻撃シナリオを表現することはできない。

対象ホスト情報格納部に蓄積される情報から、別の情報を推論するた

めの推論手段がない。これは、例えば対象ホストのＯＳがＵＮＩＸ（登録商標）であることから管理者のアカウント名がｒｏｏｔであるという知識を導出するための手段である。従って、各検査実行手段には、必要とする情報を、蓄積されている情報から推論するためのロジックを埋め込まなければならない。

攻撃者はあるホストへの侵入に成功すると、そこを踏み台としてさらに内部への侵入を試みる場合が多い。しかし従来の検査システムでは検査実行手段から直接検査を行うため、踏み台の使用を用いた検査シナリオを実施できない。

本発明は上記の問題を解決するためになされたものであり、以下の事柄を目的とする。

検査シナリオを、プログラミング言語で記述されたスクリプトとして表現し、スクリプトから自動的にプラグイン（検査実行手段に該当）を呼び出すことで、複雑な試験の実施を可能とする。

各検査実行手段間のパラメータの授受はスクリプトが媒介するようにすることで、使用者は検査実行手段間の入出力の関係について知っている必要を無くす。

セキュリティホール診断を行う際に、より現実に近い高度な攻撃シナリオに基づいた検査を実施することを可能とし、使用者に必要とされるセキュリティの知識の程度を軽減することができ、検査ロジックの作成者の負担を軽減する。

発明の開示

本発明に係るセキュリティホール診断システムは、不正アクセスのために通常攻撃者が行う手順がプログラミング言語で記述されたスクリプトが複数蓄積されたスクリプト蓄積部と、

利用者からの入力により上記スクリプトの一覧を要求する操作部と、

上記操作部の要求に応じ、上記スクリプト蓄積部から各スクリプトを取り出し、入出力パラメータ記述、スクリプト実行必要条件、検査手順を表示したリストを作成して利用者に提示し、利用者が選択したスクリプトを実行するスクリプト制御部と、

個々のセキュリティホール攻撃のためのロジックが実装されたプラグインが蓄積されたプラグイン蓄積部と、

スクリプト制御部がスクリプトを実行することにより呼び出され、上記プラグイン蓄積部から上記実行スクリプトに対応するプラグインを取り出して、そのプラグインを検査対象コンピュータに対して実行するプラグイン制御部

とを備えた。

図面の簡単な説明

図１は、実施の形態１に係るセキュリティホール診断システムの概略構成図。

図２は、図１に示す脆弱性検査装置の内部構成図。

図３は、図１に示す踏み台模擬プログラムの内部構成図。

図４は、スクリプトの構成説明図。

図５は、スクリプト制御部の動作流れ図。

図６は、クラス名を指定して検査を実行する場合の動作流れ図。

図７は、知識ファイルの例を示す説明図。

図８は、スクリプトの記述例を示す説明図。

図９は、従来のセキュリティホール診断システムを示す構成図。

図１０は、従来のシステムにおける手順定義ファイルの説明図。

図１１は、従来のシステムにおける検査実行部の情報（検査実行情報

）の説明図。

発明を実施するための最良の形態

実施の形態１．

はじめに図１を参照しながら、本システムの概要について述べる。本システムはローカルで動作する脆弱性検査装置１００とリモートもしくはローカルのホストコンピュータである１つ以上の踏み台模擬装置から構成される。本実施の形態においては１０５０、１０６０の二つの踏み台模擬装置が配置され、脆弱性検査装置１００と踏み台模擬装置１０５０、１０６０はネットワークで接続されている。また、踏み台模擬装置１０５０、１０６０は、それぞれ踏み台模擬プログラム１０５、１０６を実行する。

脆弱性検査装置１００は、使用者１０１からの要求に応じて、対象となるホストコンピュータ又は、ネットワークに対してセキュリティ上の脆弱性があるかどうかを検査する計算機である。検査は脆弱性検査装置１００が踏み台模擬装置１０５０の踏み台模擬プログラム１０５を操作することで実施される。

踏み台模擬装置１０５０が実行する踏み台模擬プログラム１０５はネットワークを通じて脆弱性検査装置１００から命令を受け取り、パケット送受信、プロセスの起動・終了、ファイル転送、メッセージ中継を行うプログラムである。

踏み台模擬プログラム１０５は他の踏み台模擬装置１０６０の踏み台模擬プログラム１０６に命令を転送する機能も有しており、踏み台模擬装置１０５０、１０６０を適切に配置することで内部ネットワークに位置する検査対象ホストコンピュータ１０７に対しても検査が行えるようになる。

踏み台模擬プログラム１０５、１０６は、検査前に検査対象のネットワーク上のホスト内で、動作させておくことも、また、脆弱性検査の一環として、セキュリティホールを利用して埋め込むことも可能である。

踏み台模擬プログラム１０５の操作は、実際には脆弱性検査装置１００内でプラグイン１０４により制御される。プラグイン１０４とは、個々のセキュリティホールを攻撃するための動的ロード可能な共有ライブラリである。プラグイン１０４は踏み台模擬プログラム１０５を操作することで検査対象上に存在するセキュリティホールへの攻撃を行う。

様々なプラグイン１０４を用意することで、多様なセキュリティホールに対する脆弱性検査が可能となる。

プラグイン１０４はスクリプト１０２によって制御される。スクリプト１０２とは、不正アクセスのために通常攻撃者が行う手順をインタプリタ言語で記述したテキストデータである。スクリプト１０２に従って様々なプラグイン１０４を呼び出すことで、脆弱性検査装置１００は、攻撃者を模擬した複雑な脆弱性検査を行うことが可能となっている。

スクリプト１０２もプラグイン１０４同様、その目的に応じて複数用意することができる。また、スクリプト１０２から他のスクリプト１０２を呼び出すことも可能であり、これにより他のスクリプト１０２を攻撃の一ステップとするような、より高度なスクリプト１０２を記述することができる。

本実施の形態では、スクリプト１０２の記述言語としてＰｅｒｌを使用している。

スクリプト１０２は、検査を実行した結果得られた検査対象に関する知識、例えばユーザアカウントの一覧や動作しているサーバ一覧等の情報を、知識共有部１０３に蓄えることが可能である。知識共有部１０３に蓄積された知識は、他のスクリプト１０２から参照することができる

また、知識共有部１０３に推論ルールに基づいて知識を吟味する推論部１０８を備えることで、スクリプト１０２から得られた知識（事実情報）から新しい知識（推論）を導出することも可能である。例えばあるスクリプト１０２によって検査対象ホストコンピュータ１０７のＯＳがＵＮＩＸ（登録商標）系であることが判れば、推論ルールにより、そのホストの管理者アカウント名がｒｏｏｔである、という知識を導出することができる。

以上の概要を踏まえた上で、次に、図２を参照しながら脆弱性検査装置１００の内部構成について説明する。脆弱性検査装置１００は、操作部２０１、検査実行部２０２で構成されており、検査実行部２０２はスクリプト制御部２０３、プラグイン制御部２０４、知識共有部１０３、及び踏み台模擬プログラム制御部２０５で構成されている。

スクリプト制御部２０３は、スクリプト１０２を蓄積・閲覧・実行するための手段を提供する。一つ以上のスクリプト１０２がスクリプト制御部２０３内にあるスクリプト蓄積部２０６に蓄積されている。スクリプト蓄積部２０６内でスクリプト１０２は、ファイル名によって一意に名前付けされ管理されている。また、スクリプト蓄積部２０６は、たとえば磁気ディスクである。

スクリプト１０２は、図４に示すとおり、クラス名記述部４０１、実行条件記述部４０２、入出力パラメータ記述部４０３、説明記述部４０４、及び検査手順記述部４０５で構成されている。

クラス名記述部４０１には、そのスクリプト１０２がどのようなカテゴリの検査に属しているかを表すデータが記述されている。実行条件記述部４０２には、スクリプト実行時に満たされていなければならない条件が記述されている。条件は述語論理を用いて記述される。入出力パラ

メータ記述部４０３には、スクリプト１０２がどのような入力を受け取り、どのような出力を行うかが記述されている。説明記述部４０４には、スクリプト１０２の説明文が記述されている。検査手順記述部４０５には検査手順が記述されている。

スクリプト１０２の記述例を図８に示した。図中、"Ｃｌａｓｓ："がクラス名記述部４０１を表し、"Ｐｒｅｃｏｎｄｉｔｉｏｎ："が実行条件記述部４０２を表し、"Ｉｎｐｕｔ："及び"Ｏｕｔｐｕｔ："が入出力パラメータ記述部４０３を表している。"Ｄｅｓｃｒｉｐｔｉｏｎ："が説明記述部４０４であり、"＃－－－－－ＥＮＤ＿ＳＣＲＩＰＴ＿ＰＲＯＰＥＲＴＹ－－－－－"　より下の部分に検査手順記述部４０５となるＰｅｒｌコードが記載される。

プラグイン制御部２０４内にはプラグイン蓄積部２０７があり、１つ以上のプラグイン１０４が蓄積されている。プラグイン蓄積部２０７は、たとえば磁気ディスクである。プラグイン１０４は、プラグイン蓄積部２０７内で一意に名前付けされて管理されている。

知識共有部１０３は、スクリプト１０２が脆弱性検査の過程で収集した知識を他のスクリプト１０２と共有することを可能とするための手段である。

知識共有部１０３内には、知識蓄積部２０８があり、脆弱性検査の過程で収集された知識が蓄積されている。知識蓄積部２０８は、たとえば磁気ディスクである。また、知識共有部１０３内には推論部１０８があり、知識蓄積部１０３内の知識を元に推論処理を行うことが可能となっている。推論処理の一環としてスクリプト制御部２０３を通じてスクリプト１０２を実行することも可能である。

踏み台模擬プログラム制御部２０５は、プラグイン１０４に対し踏み台模擬プログラム１０５を制御するためのインタフェースを提供すると

ともに、稼動中の踏み台模擬プログラム１０５の状態管理も行う。

なお、脆弱性検査装置１００は、例えばマイクロプロセッサ等のＣＰＵ、半導体メモリ等や磁気ディスク等の記録手段、及び通信手段を有する計算機により実現することができる。図２に示した知識共有部１０３、スクリプト制御部２０３、プラグイン制御部２０４及び踏み台模擬プログラム制御部２０５をプログラム（脆弱性検査プログラム）とし、記録手段に脆弱性検査プログラムを格納し、ＣＰＵが脆弱性検査プログラムを読み込むことにより脆弱性検査装置１００の動作を制御し、以下に示す処理を実現するようにしてもよい。

次に、図３を参照しながら図１中の踏み台模擬装置１０５０が実行する踏み台模擬プログラム１０５の内部構成について説明する。踏み台模擬プログラム１０５は、全体制御部３０１、通信中継部３０２、検査パケット送受信部３０３、プロセス実行部３０４及びファイル転送部３０５で構成されている。通信中継部３０２は、ネットワークを通じて、他の踏み台模擬装置１０６０の踏み台模擬プログラム１０６や図２に示す踏み台模擬プログラム制御部２０５と通信を行う。

全体制御部３０１は、通信中継部３０２を通じて送られてきた制御メッセージを受け取り、その指示に従って検査パケット送受信部３０３、プロセス実行部３０４、ファイル転送部３０５を操作する。また、制御メッセージが自分宛で無い場合には通信中継部３０２を利用して、制御メッセージを本当の宛先に転送する。

通信中継部３０２は制御メッセージを転送する。通信中継部３０２は、一つの親と複数の子と接続可能である。そのため、踏み台模擬装置１０５０は、脆弱性検査装置１００を頂点としたツリー状に相互接続される。

接続は、ＴＣＰによって行われ、ＴＣＰ接続要求は子から親、親から

子どちらからも可能である。

次に図２を用いて本システムの動作について説明する。

はじめに使用者１０１は操作部２０１を通じて、検査実行部２０２に対し、実行可能なスクリプト１０２の一覧を要求する。検査実行部２０２はその内部手段であるスクリプト制御部２０３を呼び出す。

スクリプト制御部２０３は、スクリプト蓄積部２０６からスクリプト１０２を一つずつ取り出し、そのファイル名、入出力パラメータ部４０３、説明記述部４０４、及びクラス名記述部４０１の内容をリストに蓄積する。全てのスクリプト１０２に対してこの処理を繰り返したら、リストを操作部２０１を通じて使用者１０１に返す。

次に、使用者１０１は検査の一覧（リスト）から自分の行いたいスクリプト１０２を選択し、操作部２０１を通じて検査実行部２０２に対し、検査の実行を要求する。要求には、（１）　スクリプト名もしくはクラス名、（２）　検査パラメータの情報、（３）検査終了条件（但し（１）がクラス名の場合のみ）が含まれている。検査実行部２０２は、スクリプト制御部２０３に対し、検査の実行を要求する。実行結果は操作部２０１に返される。

次に図２、図４、図５を参照しながらスクリプト制御部２０３の動作について説明する。はじめに検査名を指定して検査を実行する場合について説明する。

検査実行要求を受けたスクリプト制御部２０３は、ステップ５０１でスクリプト蓄積部２０６内に指定されたファイル名で管理されたスクリプト１０２を取り出す。

次にステップ５０２で、スクリプト制御部２０３はスクリプト１０２に記載されている実行条件記述部４０２の内容を取り出す。スクリプト１０２の実行条件記述部４０２には、そのスクリプト１０２を実行する

ために必要な条件、例えば検査対象ホストコンピュータ１０７のＯＳがＷｉｎｄｏｗｓ（登録商標）であること等が述語論理で記述されている。スクリプト制御部２０３は、この条件を知識共有部１０３に渡し、実行条件が満たされているかどうかを確認する。

次に知識共有部１０３からの応答を元に、実行条件が満たされたかどうかの判断をステップ５０３で行い、もし実行条件が満たされていなければスクリプト制御部２０３は、ステップ５０８に進みスクリプト１０２の実行失敗として処理を終了する。

もし実行条件が満たされていたならば、処理はステップ５０４に進む。ここでスクリプト制御部２０３は、スクリプト１０２の検査手順記述部４０５の内容と、検査実行要求に含まれる検査パラメータに従い、検査を実行する。

ステップ５０５でスクリプトの実行結果が判断され、失敗した場合はステップ５０８に進み、処理を終了する。

実行に成功した場合、新たな知識が獲得される場合がある。例えば、発見されたセキュリティホールの一覧等である。そのような知識は他の検査を行うときに再利用できるようステップ５０６で、知識共有部１０３中の共有知識蓄積部２０８中に格納しておく。

最後に、実行結果を呼び出し元に返して処理は終了する（ステップ５０７）

次に、図６を参照しながらクラス名を指定して検査を実行する場合について説明する。

検査実行要求を受けたスクリプト制御部２０３は、ステップ６０１～ステップ６０７で構成されるループを実行することで、スクリプト蓄積部２０６中に格納されているスクリプト１０２を順に取り出し、以下の動作を行う。

まず、ステップ６０４で現在対象としているスクリプト１０２のクラス名記述部４０１を参照し、そのスクリプト１０２が検査実行要求で指定されたクラスに所属しているかどうかを検査する。

もしスクリプト１０２が検査実行要求で指定されたクラス１０２に所属していなければ、ステップ６０９に進み、次のスクリプト１０２に対して、処理を行う。

スクリプト１０２が検査実行要求で指定されたクラスに所属していたならば、ステップ６０５で、スクリプト１０２の実行を試みる。具体的には、図５のステップ５０２からの処理を行うことになる。

ステップ６０６で実行の成功・失敗を判断し、もし失敗したならば、ステップ６０９に進み、他のスクリプト１０２の実行を試みる。

実行が成功した場合、さらに他の同一クラスのスクリプト１０２を実行するかどうかをステップ６０７で判断する。判断は、検査実行要求として渡される情報に含まれる、検査終了条件をもとに行われる。

もし、検査終了条件が、「クラスが一致する全てのスクリプトを実行」であったならば、ステップ６０９に進み、他のスクリプト１０２についても実行を試みる。そうでなければステップ６０８に進み、実行結果を呼び出し元に返して処理は終了する。

ステップ６０２で、全てのスクリプト１０２に対して実行を試みたかどうかが判定され、もし全てのスクリプト１０２に対して実行を試みたことが判明した場合には、処理はステップ６１０に進む。

ステップ６１０に到達するまでに、一つでもスクリプト１０２の実行に成功していた場合には、ステップ６０８に進み、実行結果を呼び出し元に返して処理は終了する。もし一つも成功していなかった場合には、ステップ６１１に進み、検査実行処理失敗として処理を終了する。

以上、使用者１０１によってスクリプト実行を要求された場合の処理

について述べたが、前述したとおり、スクリプト１０２から他のスクリプト１０２を呼び出すことも可能である。この場合、呼び出し元が異なるだけで、スクリプト制御部２０３に渡すデータ及びその後の処理は同一である。

次に、図２を参照しながらプラグイン制御部２０４の動作について説明する。プラグイン制御部２０４は、スクリプト１０２の検査手順記述部４０５に記述されたプラグイン実行命令をスクリプト制御部２０３が実行した時にスクリプト制御部２０３によって呼び出される。呼び出し時に渡されるデータは実行するプラグイン１０４の名前及びそのプラグイン１０４が必要とする実行パラメータである。

プラグイン制御部２０４はプラグイン蓄積部２０７から、パラメータとして渡されたプラグイン名に対応するプラグイン１０４を取り出して実行する。実行結果は呼び出し元であるスクリプト制御部２０３に返され、最終的にはプラグイン実行命令に対する結果としてスクリプト１０２に戻される。

プラグイン１０４はその実行中に、踏み台模擬プログラム制御部２０５を通じて、踏み台模擬プログラム１０５を操作する。操作される踏み台模擬プログラム１０５は、プログラムの動作しているホストコンピュータのアドレスと、ホストコンピュータ内部でユニークな踏み台模擬プログラム識別子で指定される。踏み台模擬プログラム１０５に要求できる命令は以下の通り。

ＴＣＰ／ＵＤＰ／ＲＡＷソケット生成・破棄

ソケット（ＴＣＰ／ＵＤＰ）のローカルポートへのＢｉｎｄ

ソケット（ＴＣＰ／ＵＤＰ）のリモートポートへのＣｏｎｎｅｃｔ

Ｃｏｎｎｅｃｔされたソケットを通じたＳｅｎｄ、　Ｒｅｃｖ

Ｃｏｎｎｅｃｔされていないソケットを通じたＳｅｎｄＴｏ、　Ｒｅｃ

ｖＦｒｏｍ

Ｐｒｏｃｅｓｓの起動・終了

起動したＰｒｏｃｅｓｓの標準入出力を通じたデータのやり取り

脆弱性検査装置ホストから踏み台模擬プログラム動作ホストへのファイル転送及びその逆踏み台模擬プログラム状態取得

踏み台模擬プログラム停止

次に図２を参照しながら知識共有部１０３の動作について説明する。知識共有部１０３は知識蓄積部２０８に、検査によって得られた知識を蓄積し、他の検査でそれを再利用することを可能にするために使用される。

推論部１０８は、与えられたゴールを満たす解が存在するかどうか、知識蓄積部２０８中の知識に基づいて推論を行う。本手段は、スクリプト１０２の実行条件の確認のためにスクリプト制御部２０３によって呼び出される。また、スクリプト１０２に共有知識獲得命令を記述しておくことで、スクリプト実行中に呼び出される場合もある。

知識は述語論理で表現されており、推論はＰｒｏｌｏｇ等の、述語論理に基づいた推論システムによって行われる。知識蓄積部２０８には、検査で得られた事実に関する知識だけでなく、変数を利用した推論ルールも蓄積しておくことが可能である。

また、スクリプト１０２を実行する作用を持った特別な述語が定義されており、この述語を利用した推論ルールを記述しておくことで、共有知識が不足の場合に知識を獲得するためにスクリプト１０２を実行することができる。これにより、あるスクリプト１０２の実行条件を満たすために、自動的に他のスクリプト１０２を呼び出すことが可能となる。

推論ルールは通常システム初期化時に初期設定ファイル（知識ファイル）から読み取られ、共有知識蓄積部２０８に設定されるが、検査の過

程で追加することも可能である。また、蓄積された知識を初期設定ファイル（知識ファイル）に保存することも可能である。

知識ファイルの例を図７に示した。本実施の形態では、記法はＰｒｏｌｏｇの文法を利用している。

本実施の形態で示されるシステムにより、次のような特徴を持ったセキュリティホール診断システムを実現することができる。

第一に検査シナリオを、プログラミング言語で記述されたスクリプト１０２として表現し、スクリプト１０２から自動的にプラグイン（検査実行部に該当）１０４を呼び出すことで、複雑な試験の実施を実施できる。

さらに、各検査実行部間のパラメータの授受はスクリプト１０２が媒介するようにすることで、使用者は検査実行部間の入出力の関係について知っている必要を無くせる。

さらに、スクリプト１０２が他のスクリプト１０２を呼び出せるようにすることで、階層化されたシナリオの実施を実現できる。

さらに、推論ルールに従って、共有された知識から新たな知識を導出できるようにすることで各スクリプト１０２・プラグイン１０４毎に推論ロジックを作りこむ必要が無くなる。

さらに、プラグイン１０４が、踏み台模擬プログラム１０５を経由して検査を実行することで、現実の攻撃者と同様な踏み台を経由した検査シナリオを実現できる。

さらに、スクリプトにクラスの概念を採り入れることにより、クラス名によるグループ分けが可能に成り、スクリプトから他のスクリプトを呼び出すときに、スクリプトのファイル名でなく、クラス名からも呼び出すことができる。

実施の形態２．

実施の形態１では、操作部２０１と検査実行部２０２は同一装置内に存在するが、これらをネットワーク上に分散配置することも可能である。

本実施の形態で示されるシステムにより、次のような特徴を持ったセキュリティホール診断システムを実現することができる。

実施の形態１における特徴に加え、検査実行部をファイアウォールの外側に配置し、操作部をファイアウォールの内側に配置することが可能となり、これにより、本システムをネットワーク上に配置することのセキュリティ上のリスクを低減することが可能である。

実施の形態３．

実施の形態１では、プラグイン１０４として動的ロード可能な共有ライブラリを用いているが、踏み台模擬プログラム制御部２０５とのインタフェースを提供可能なインタプリタ言語によっても実現可能である。

本実施の形態で示されるシステムにより、次のような特徴を持ったセキュリティホール診断システムを実現することができる。

実施の形態１における特徴に加え、プラグイン１０４をより実装しやすくなる上、システム運用中でも簡単にプラグイン１０４を編集可能となる。

実施の形態４．

本実施の形態では踏み台模擬プログラム１０５、１０６同士、及び踏み台模擬プログラム１０５と脆弱性検査装置１００との間の通信はＴＣＰ／ＩＰ上の独自プロトコルを用いたが、ファイアウォールを考慮してこれをＨＴＴＰ、　ＳＭＴＰ等のファイアウォール通過可能な一般的通

信プロトコル上に構築することも可能である。

本実施の形態で示されるシステムにより、次のような特徴を持ったセキュリティホール診断システムを実現することができる。

実施の形態１における特徴に加え、踏み台模擬プログラムとの通信がファイアウォールによって遮断されることを防ぐことができ、より実際の攻撃者と同等の攻撃シナリオで検査を行えるようになる。

産業上の利用可能性

以上述べたように本発明によれば、検査シナリオを、プログラミング言語で記述されたスクリプトとして表現し、スクリプトから自動的にプラグイン（検査実行部に該当）を呼び出すことで、複雑な試験を実施できる。

さらに、各検査実行部間のパラメータの授受はスクリプトが媒介するようにすることで、使用者は検査実行部間の入出力の関係について知っている必要を無くせる。

請求の範囲

1．不正アクセスのために通常攻撃者が行う手順がプログラミング言語で記述されたスクリプトが複数蓄積されたスクリプト蓄積部と、

利用者からの入力により上記スクリプトの一覧を要求する操作部と、

上記操作部の要求に応じ、上記スクリプト蓄積部から各スクリプトを取り出し、入出力パラメータ記述、スクリプト実行必要条件、検査手順を表示したリストを作成して利用者に提示し、利用者が選択したスクリプトを実行するスクリプト制御部と、

個々のセキュリティホール攻撃のためのロジックが実装されたプラグインが蓄積されたプラグイン蓄積部と、

スクリプト制御部がスクリプトを実行することにより呼び出され、上記プラグイン蓄積部から実行スクリプトが指定するプラグインを取り出して、そのプラグインを検査対象コンピュータに対して実行するプラグイン制御部

とを備えたセキュリティホール診断システム。

2．パケット送受信、プロセス起動・終了・プロセスとのデータ入出力、ファイル転送機能を有する踏み台模擬プログラムと、上記プラグインからの指令により検査対象コンピュータに対するプラグインの実行を上記踏み台模擬プログラムを介して実施する踏み台模擬プログラム制御部とを備えたことを特徴とする請求項1記載のセキュリティホール診断システム。

3．上記スクリプトは他のスクリプトを呼び出せる機能を有するように構成されたことを特徴とする請求項1に記載のセキュリティホール診断システム。

4．上記スクリプトにクラスの概念が導入され、上記スクリプトは、他のスクリプトを呼び出す際、クラス名を指定することにより他のスクリプトを呼び出せる機能を有するように構成されたことを特徴とする請求項１記載のセキュリティホール診断システム。

5．上記スクリプト実行必要条件が満たされているか否かを確認する知識共有部を備え、知識共有部は上記スクリプトが実行される過程で収集された情報を推論ルールに従って新しい知識に導出する推論部を有することを特徴とする請求項１記載のセキュリティホール診断システム。

6．知識共有部は、共有知識が不足の場合に、推論ルールに従って知識獲得のためのスクリプトを実行する機能を有するように構成されたことを特徴とする請求項５記載のセキュリティホール診断システム。

7．上記スクリプト制御部と、上記プラグイン蓄積部と、上記プラグイン制御部と、上記スクリプト蓄積部と、上記踏み台模擬プログラム制御部とで検査実行部を形成し、検査実行部と上記操作部とはネットワーク上に分散した構成とされた請求項２記載のセキュリティホール診断システム。

8．上記プラグインはインタプリタ言語で記述されたものであることを特徴とする請求項１記載のセキュリティホール診断システム。

9．踏み台模擬プログラム制御部は、ファイアウォールが通過可能なプロトコル上で構築されていることを特徴とする請求項２記載のセキュリティホール診断システム。

要　約　書

通常攻撃者が行う手順をプログラミング言語で記述されたスクリプトが予め蓄積され、その蓄積されたスクリプトから使用者が選択したスクリプトを実行することにより、個々のセキュリティホール攻撃のためのロジックが実装されたプラグインを呼び出し、このプラグインを検査対象コンピュータに対して実行することで、使用者は検査実行部間の入出力の関係等、セキュリティ上の知識を知っている必要が無くせる。

図 1

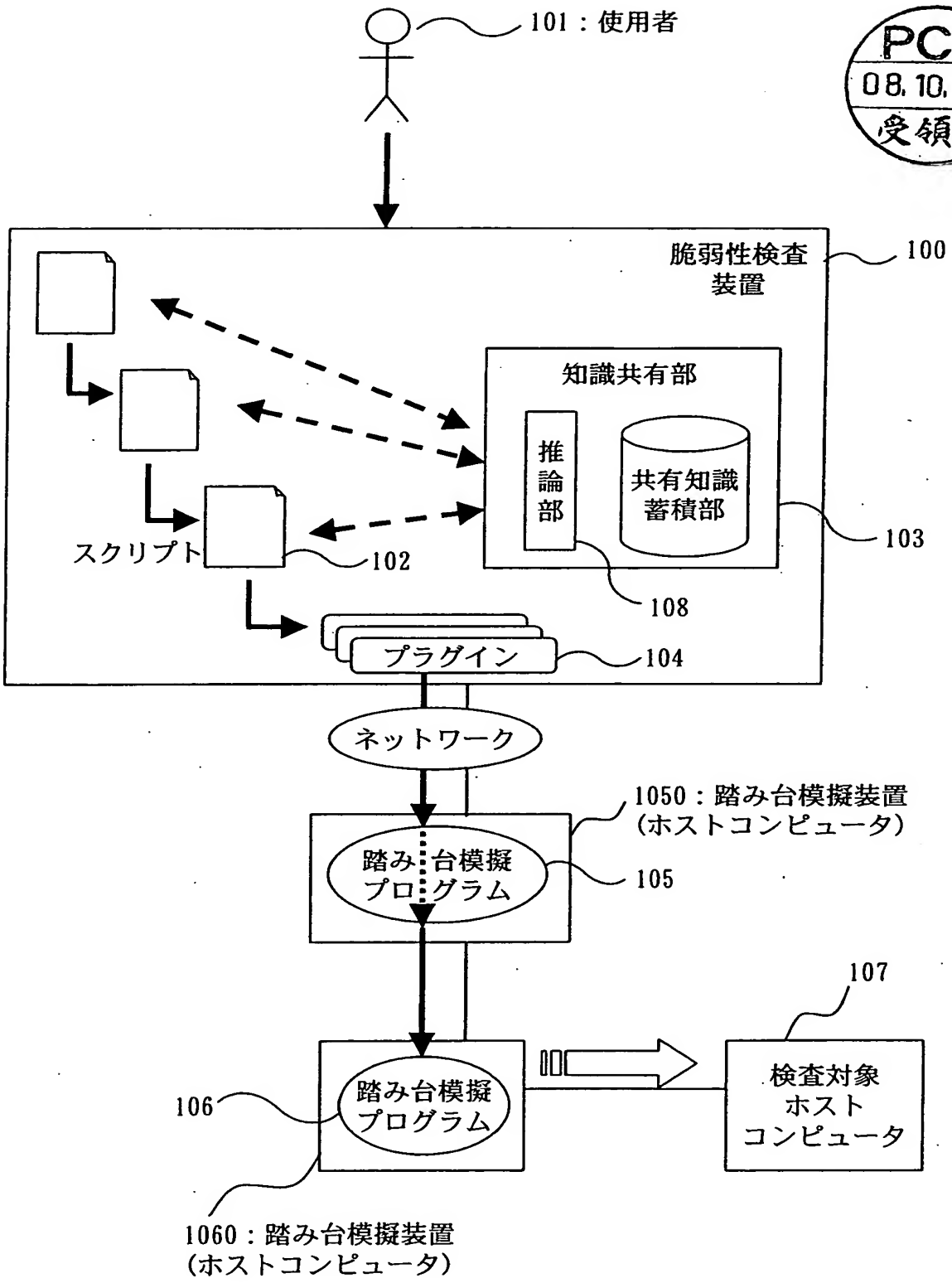

101：使用者
PCT
08.10.03
受領印
脆弱性検査
装置
100
知識共有部
推
論
部
共有知識
蓄積部
103
スクリプト
102
108
プラグイン
104
ネットワーク
1050：踏み台模擬装置
（ホストコンピュータ）
踏み台模擬
プログラム
105
107
踏み台模擬
プログラム
106
検査対象
ホスト
コンピュータ
1060：踏み台模擬装置
（ホストコンピュータ）

図 2

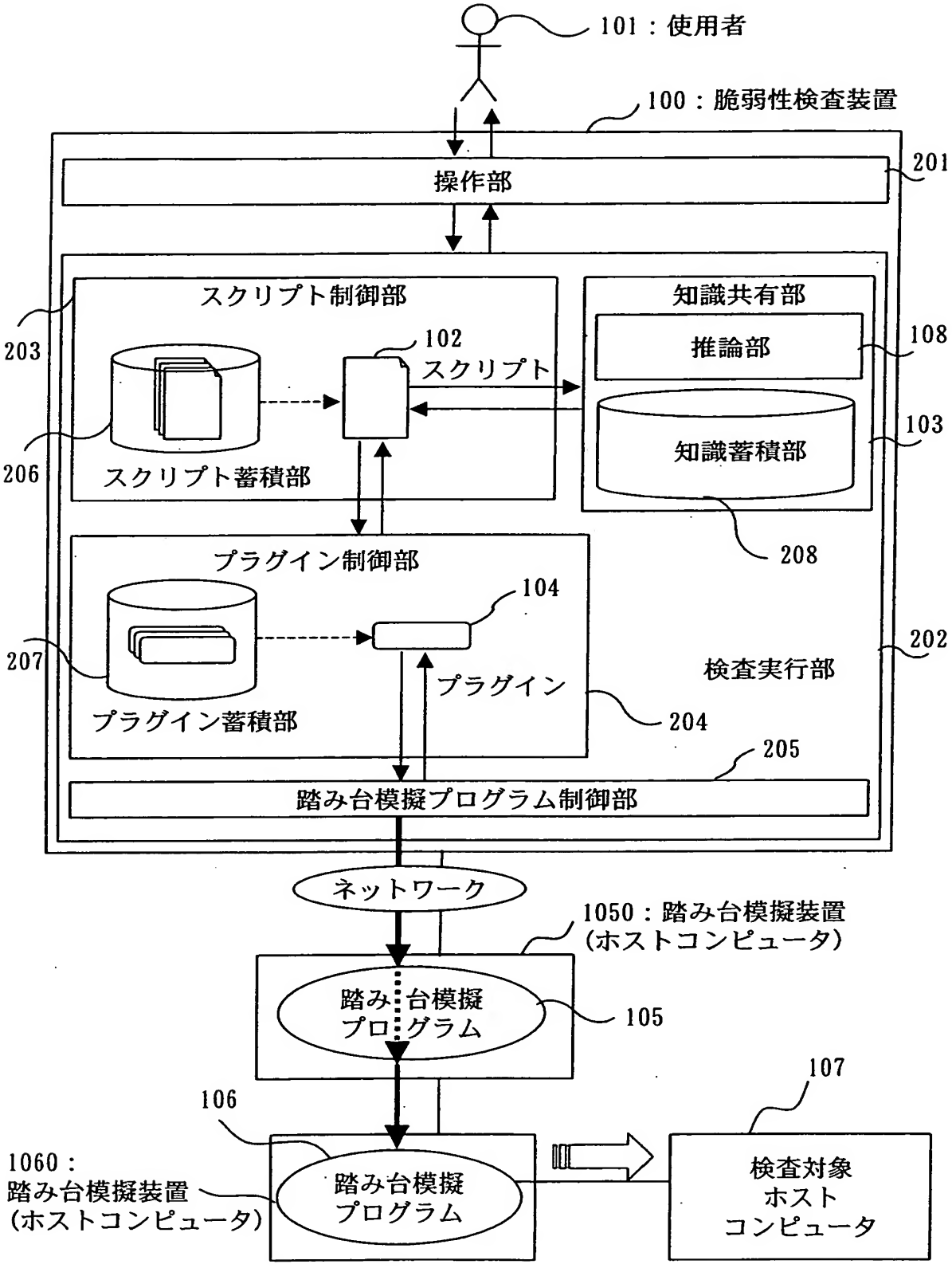

101：使用者
100：脆弱性検査装置
操作部
201
スクリプト制御部
203
102
スクリプト
206
スクリプト蓄積部
知識共有部
推論部
108
知識蓄積部
103
208
プラグイン制御部
104
207
プラグイン
プラグイン蓄積部
204
検査実行部
202
205
踏み台模擬プログラム制御部
ネットワーク
1050：踏み台模擬装置
（ホストコンピュータ）
踏み台模擬
プログラム
105
107
106
1060：
踏み台模擬装置
（ホストコンピュータ）
踏み台模擬
プログラム
検査対象
ホスト
コンピュータ

図 3

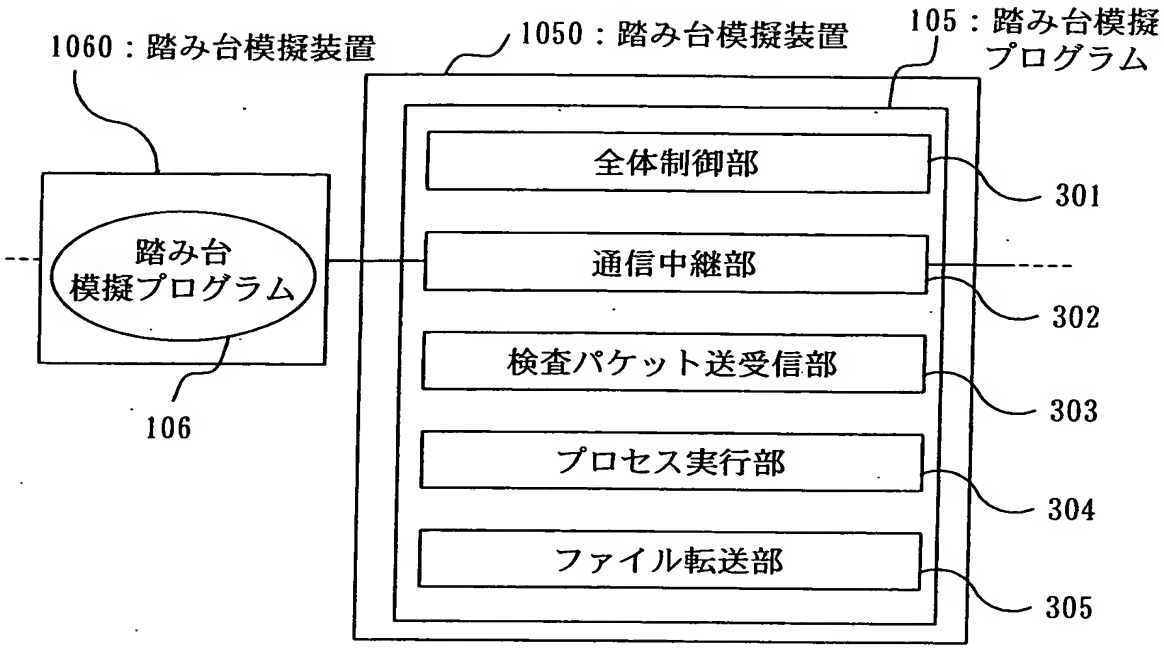

1060：踏み台模擬装置
1050：踏み台模擬装置
105：踏み台模擬
プログラム
踏み台
模擬プログラム
106
全体制御部
301
通信中継部
302
検査パケット送受信部
303
プロセス実行部
304
ファイル転送部
305

図4

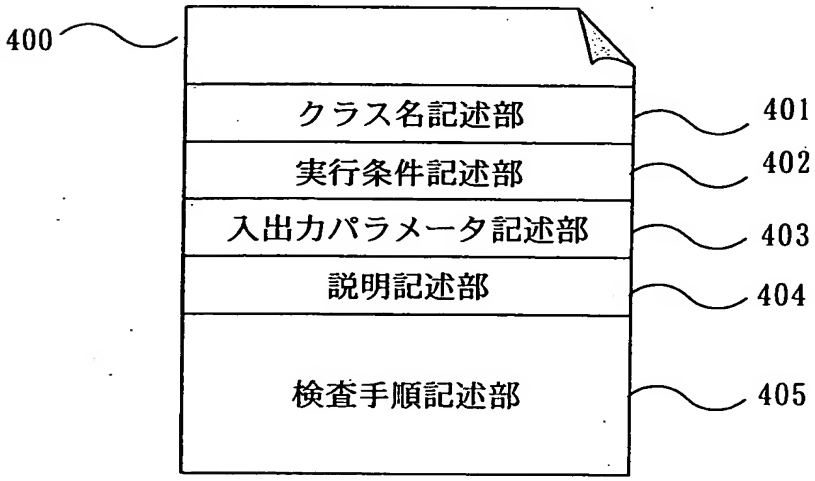

400
クラス名記述部
401
実行条件記述部
402
入出力パラメータ記述部
403
説明記述部
404
検査手順記述部
405

図 5

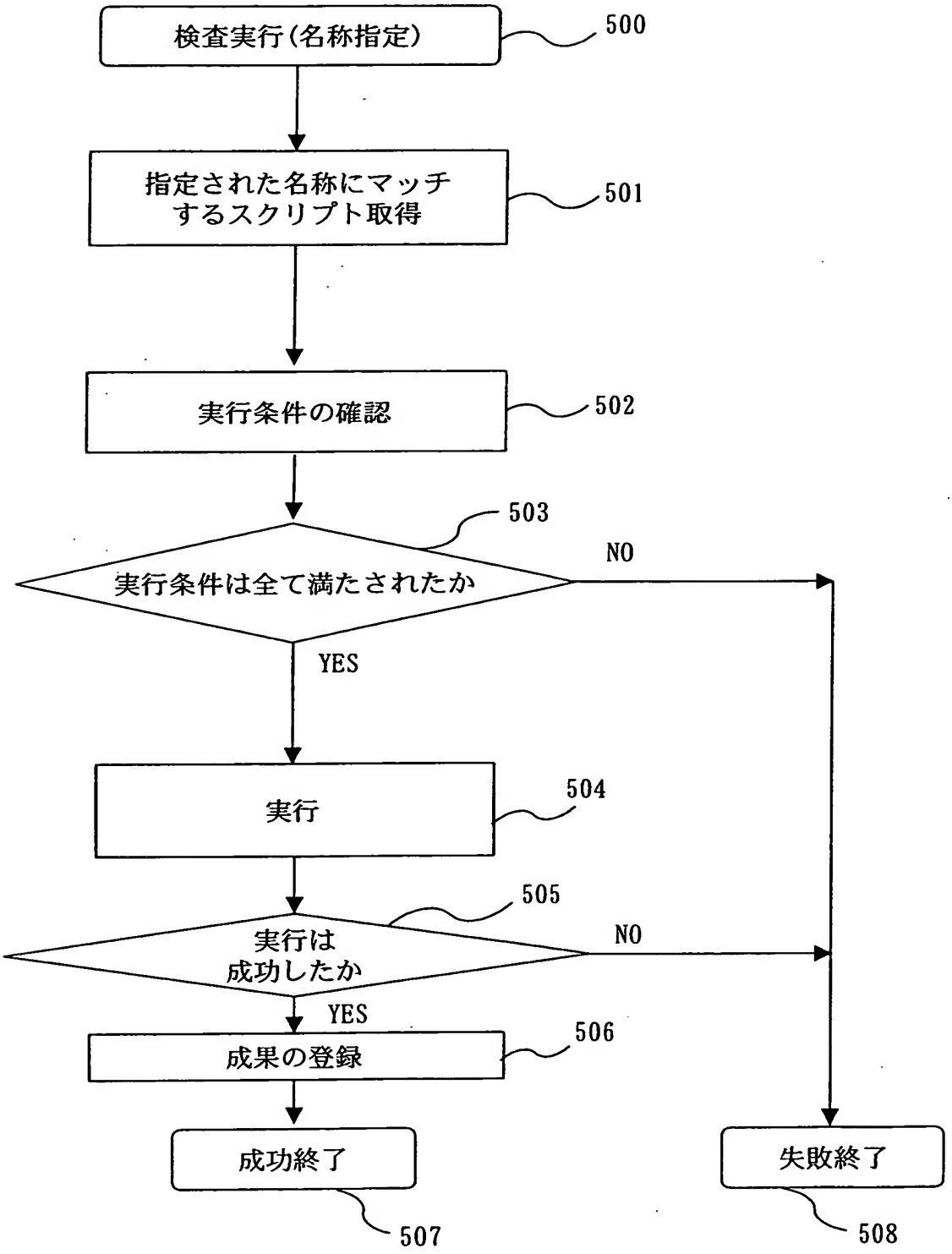

検査実行(名称指定)
500
指定された名称にマッチ
するスクリプト取得
501
実行条件の確認
502
503
実行条件は全て満たされたか
NO
YES
実行
504
505
実行は
成功したか
NO
YES
成果の登録
506
成功終了
507
失敗終了
508

図 6

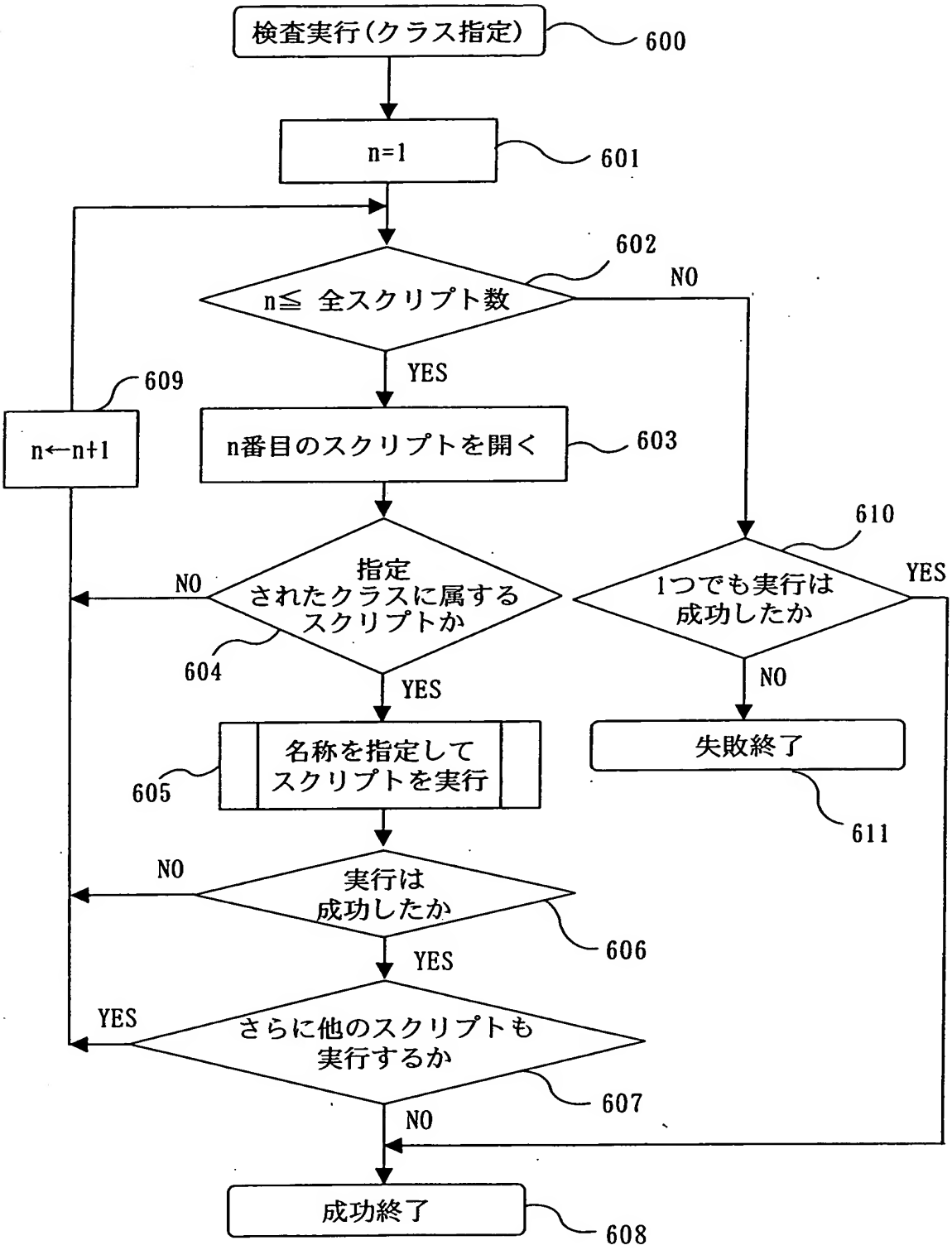

検査実行(クラス指定)
600
n=1
601
602
n≦ 全スクリプト数
NO
YES
609
n←n+1
n番目のスクリプトを開く
603
610
指定
されたクラスに属する
スクリプトか
NO
YES
1つでも実行は
成功したか
YES
604
NO
YES
NO
名称を指定して
スクリプトを実行
失敗終了
605
611
NO
実行は
成功したか
606
YES
YES
さらに他のスクリプトも
実行するか
607
NO
成功終了
608

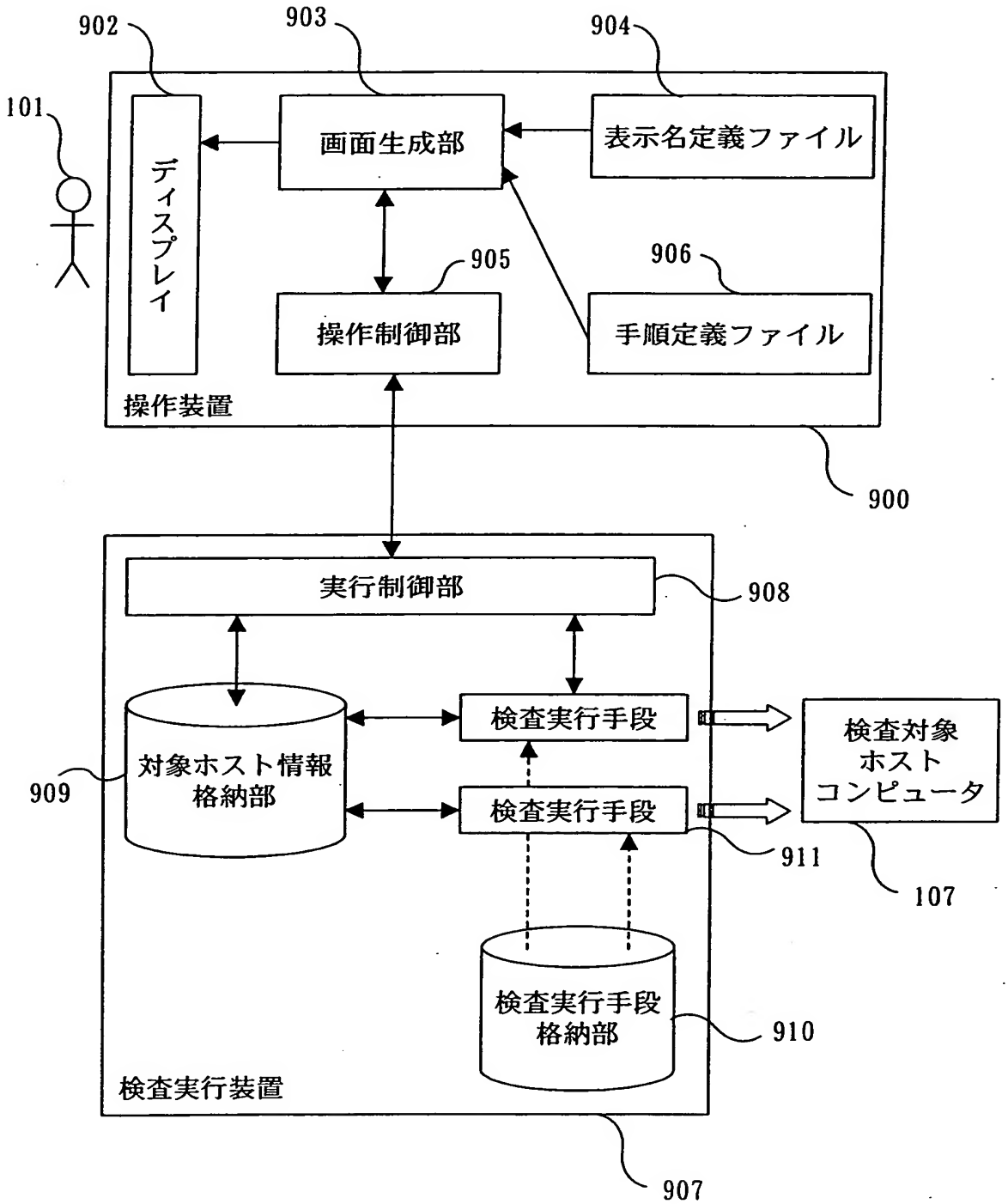
図 7

```
admin( HOST, 'root' ) :-
            os( HOST, 'UNIX' );

admin( HOST, 'administrator' ) :-
            os( HOST, 'WINDOWS' ).

web(HOST, PORT) :-
            script( 'find_web_port', [ HOST ], [PORTS] )., 
            member( PORT, PORTS ).

cgi(HOST, PORT, CGINAME) :-
            script( 'check_cgi', [ HOST, PORT, CGINAME ], _ ).
```

図8

```
# -----BEGIN_SCRIPT_PROPERTY-----
#Class:      command_exec
#Pre-condition:          web(HOST, PORT), cgi( HOST, PORT, "phf" )
#Input:      HOST        string  "検査対象ホストアドレス"
#Input:      CMD         string  "実行するコマンド文字列"
#Output:     RESULT      string  "コマンド実行出力"
#Description:            "PHFを利用してコマンドを実行します. "
#-----END_SCRIPT_PROPERTY-----

Perlによる実行コード. . .
```

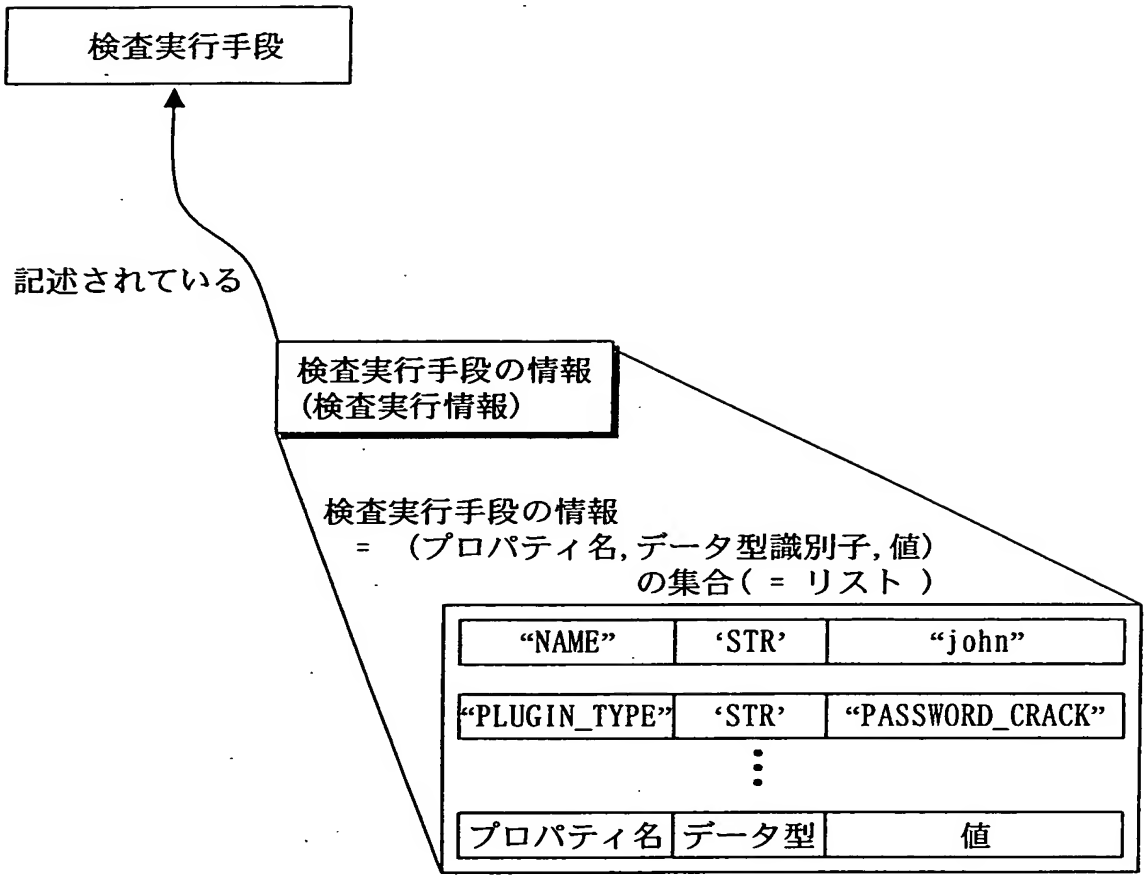
図 9


902
903
904
101
ディスプレイ
画面生成部
表示名定義ファイル
905
906
操作制御部
手順定義ファイル
操作装置
900
実行制御部
908
検査実行手段
検査対象
ホスト
コンピュータ
対象ホスト情報
格納部
909
検査実行手段
911
107
検査実行手段
格納部
910
検査実行装置
907

```
#
# 手順定義 File.
#

# PROPERTY NAME
PLUGIN_TYPE

#
# SYNTAX:  CATEGORY_KEY'='CATEGORY_DISPLAY_NAME(' 'EXEC_TYPE)(' 'CATEGORY_DESCRIPTION)
#          EXEC_TYPE = O(ne)/A(ll)/Q(uery)

PORTSCAN=ﾎﾟｰﾄｽｷｬﾝ Q 対象ﾎｽﾄ上の攻撃可能なﾎﾟｰﾄを調べます。

GET_FILE=ﾌｧｲﾙ取得 O ﾀｰｹﾞｯﾄから指定されたﾌｧｲﾙを取得します。

PASSWORD_CRACK=ﾊﾟｽﾜｰﾄﾞｸﾗｯｸ Q 指定されたﾊﾟｽﾜｰﾄﾞﾌｧｲﾙからｱｶｳﾝﾄとﾊﾟｽﾜｰﾄﾞを抽出します。
```

図10

図11


検査実行手段
記述されている
検査実行手段の情報
(検査実行情報)
検査実行手段の情報
= (プロパティ名, データ型識別子, 値)
の集合(= リスト)
"NAME"
'STR'
"john"
"PLUGIN_TYPE"
'STR'
"PASSWORD_CRACK"
プロパティ名
データ型
値